# Quadruped Robot Locomotion with Reinforcement Learning

**GROUP 18:** HO KAE BOON (A0219811H), HUANG WENBO (A0285085W), ZENG ZHENGTAO (A0285134E), ZHONG HAO (A0284914W)

**Why:**

• Traditional wheeled and tracked robots are limited in environments, which are full of irregular terrains. Quadruped robots, in comparison, have greater advantages when exploring complicated environments. In fact, the recent development of quadruped robots has seen its application in disaster relief, fighting against terrorism and so on. Thus, developing better control algorithms for the robots to walk smoothly and stably in various terrains is important to enable the quadruped robots to finish their work. In this project, we would like to start from the basics, that is, to train the quadruped to walk on a planar terrain. The approach could easily be extended to unstructured terrain with obstacles.

**Conventional Algorithms:**

In conventional algorithms, such as inverse kinematics and gait algorithms, which calculate the optimal joint values from end-effector positions, debugging and maintenance are much easier to handle due to the widespread availability of the algorithm's resources. However, these algorithms have some drawbacks, including the difficulty of non-linear controller design, various constraints and typically ruled-based systems that maintain more complexity than reinforcement learning. Furthermore, conventional algorithms are mainly aimed to be applied to predefined simple environments instead of unstructured environments with various obstacles.

In comparison, in reinforcement learning methods, quadruped robots can develop intelligent algorithmic control strategies through continuous trial and error, instead of by human design. The agent can learn quadruped locomotion independently by interacting with the environment, enabling the ability to learn and update their strategies to navigate different terrain conditions. Thus, we believe using reinforcement learning for developing quadruped locomotion strategies is beneficial, as it can help to save time and trouble when designing the locomotion controller from scratch using kinematics, and allow a more flexible controller to be developed.

**Problem Statement:**

• (How robots know about systems): In this project, a physics simulator such as PyBullet will be used as the main environment. A planar ground will be generated. Then, a quadruped digital double (with two joints on each limb, eight joints in total) will be imported to the simulator to interact with the ground;
• (What the robot know about the system): The robot will only know proprioceptive data, more details are provided in the state space below;
• (Goal): We will provide the robot with a unit vector pointing to the front of the robot’s initial frame. The robot will be tasked to walk towards the front direction while maintaining a constant velocity.
• (Capabilities): The robot's eight joints can exert torque to rotate for a discretized angle value.

**RL Cast:**

• State space:
The robot state will be the robot’s proprioceptive measurements, directly obtained from the simulation,

This includes:
a) Robot torso's centre of mass pose (position + orientation);
b) Angular position of each link for each leg;
c) Intersection of volumes for each robot link (For the detection of collision);
d) Torque for each joint from the previous time step.

• Termination state:
The episode terminates during the following states:
a) The robot torso's centre of mass from the ground is low (say below 0.3m);
b) Any collision between the robot's links;
c) The joint limits are exceeded;
d) The robot deviates from the given direction largely (say 10 meters);
e) The robot does not accelerate after a certain period (say 10 seconds).

• Action space:
Discretized +/= 10 Nm eight joint torque signals for each rotational joint to produce a walking gait pattern for robot feet. This results in 2^8 = 256 possible actions for each state transition.

• Reward structure:
a) Velocity of robot torso's centre of mass towards the walking direction: Big positive reward for positive velocity towards the target direction (say +10);
b) Height of the torso from the desired height, and pitch angle of the torso: We want the robot to maintain an upright posture when walking. A large penalty will be imposed when the robot deviates from an upright posture (say -8), that is, when the height of the torso is lower than ideal (say <0.65m), and when the pitch angle of the torso exceeds say +/- 15 deg;
c) Foot slip penalty: We impose a small penalty (say -3) on foot slip, that is, foot tip velocity should be zero when in contact with the ground, to prevent slipping;
d) Joint pose penalty: We impose a penalty (say -2) on large joint angles (say +/- 60 deg) to prevent large angle displacement.
e) Jumping penalty: We impose a small penalty (say -3) when none of the feet is in contact with the ground to prevent the robot from hopping.

• Neural Network Structure:
We will employ a multi-layered neural network structure. This neural network would be incorporated into the reinforcement learning algorithm.

**RL algorithms and envisioned results:**

• We are considering using an actor-critic reinforcement learning agent, such as Deep Deterministic Policy Gradient (DDPG), which is commonly used for robotics control tasks due to its effectiveness in handling continuous states. We will conduct more studies into it. By implementing these algorithms, we expect to allow the robot to estimate the optimal locomotion policy completely by itself.